2002

A new numerical approach for solving higher order nonlinear ordinary differential equations

Hung Thanh Phan
University of Wollongong

A new numerical approach for solving higher order nonlinear ordinary differential equations

A thesis submitted in fulfillment of the

requirements for the award of the degree of

Master of Science (Honours)

from

University of Wollongong

by

Hung Thanh Phan, B.Sc (UNSW)

School of Mathematics and Applied Statistics

2002

This Thesis is submitted to The University of Wollongong and has not been submitted for a higher degree to any other University or Institution.

Hung Thảnh Phan

August, 2002

ACKNOWLEDGEMENTS

There are a lot of people who assisted me in preparing this thesis, and whom I would like to gratefully acknowledge.

First of all, I wish to express my sincere thanks to my supervisor, Associate Professor Songping Zhu, for his effort in arranging my enrolment into a master program, having great discussions with me, the care and encouragement he gave me to carry out this study.

I also would like to thank all staff and fellow friends in the School of Mathematics and Applied Statistics, particularly Prof. David Steel, and Prof. Philip Broadbridge, for their help in organizing my master study program. My thanks extend to Ms Carolyn Silveri, who has been kindly helping me through out the study, and Prof. Jim Hill for his encouragements. I would also like to thank Dr.Grant Cox in helping me in Maple software.

The completion of this thesis makes a wonderful dream of mine become true. This could not have been possible without the help of my supervisor, Associate Prof. Songping Zhu.

Abstract

There have been many numerical solution approaches to ordinary differential equations in the literature. However, very few are effective in solving non-linear ordinary differential equations (ODEs), particularly when they are of order higher than one. With the modern symbolic calculation packages, such as Maple and Mathematica, being readily available to researchers, we shall present a new numerical method in this thesis. Based on the repeated use of a symbolic calculation package and a second order finite difference scheme, our method is particularly suitable for solving higher order nonlinear differential equations arising from initial value problems. One important feature of our approach is that if the highest order derivative in an ODE can be written explicitly in terms of all the other terms of lower orders, our method requires no iterations at all. On the other hand, if the highest order derivative in an ODE can not be written explicitly in terms of all the other lower-order terms, iterations are only required before the actual time marching begins.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In many mathematical modeling and engineering problems, ordinary differential equations need to be solved numerically. Although, there have been many numerical solution approaches to the ordinary differential equations in the literature, very few of these approaches are effective in solving non-linear ordinary differential equations, particularly when they are of order higher than one. The most popularly used methods are Runge-Kutta method [1, 2] and Predictor-Corrector method based on Adams-Bashforth and Adams-Moulton methods [3, 4] or their various variations [5, 6, 7, 8].

With the development of the modern symbolic calculation packages, such as Maple and Mathematica, many mathematical problems can now be solved with these powerful packages. In fact, the wide availability of these packages to scientists and engineers prompts us to search for new methods, based on which problems previously regarded as difficult can

now be solved more efficiently and/or accurately. In this thesis, we shall propose a new numerical method for solving higher order nonlinear differential equations arising from initial value problems. Our method must be used in conjunction with a symbolic calculation package and it can be used to efficiently and accurately solve an nth-order nonlinear ordinary differential equation (ODE) arising from an initial value problem. When the highest order derivative in the ODE can be written explicitly in terms of all the other terms of lower orders, our method requires no iterations at all. On the other hand, if the highest order derivative in the ODE can not be written explicitly in terms of all the other lower-order terms, iterations are only required before the time marching begins and no iterations are required during the time marching process. This feature certainly has a great advantage over traditional approaches as far as the efficiency is concerned.

As far as the accuracy is concerned, we shall theoretically show that the difference scheme we proposed is a second order one, i.e., it is of $O(h^2)$, where h is a measure of the grid spacing. Through several numerical examples, we shall also show that for all the test problems we have carried out calculation for, the maximum relative error is much less than 1%, an accuracy that can be comfortably accepted in engineering calculations.

The thesis is divided into four sections. In Section 2, a detailed description of our new algorithm is provided. In Section 3, three numerical examples of highly nonlinear initial value problems are given and our

conclusions are stated in Section 5.

Chapter 2

Method

A general initial value problem can be written in the form

$$f(y^{(n)}(x), y^{(n-1)}(x), \ldots, y'(x), y(x), x) = 0, \tag{2.1}$$

subject to the initial conditions

$y(x_0) = a_0$, $y'(x_0) = a_1$, ..., $y^{(n-1)}(x_0) = a_{n-1}$. In Eq. (2.1), superscripts or primes are used interchangeably to denote derivatives, f is an arbitrary function of the independent variable x, the unknown function y and its derivatives up to the order of n. a_i (with $i = 1, \ldots, n-1$) are given values of the corresponding derivatives at an initial value x_0. Without loss of generality, we shall from now on assume that $x_0 = 0$ unless stated otherwise. Since in most of initial problems the independent variable usually represents time, we shall sometimes refer to the independent variable as time (e.g., the grid spacing in x is sometimes referred to as timestep, etc.).

When f is a linear function of the unknown y and its derivatives, the numerical solution approach is rather trivial; there have been many numerical methods that are often used [9]. Our new approach is particularly useful when Eq. (2.1) is nonlinear since there will be no iterations required (or at most iterations are only required in the process of transforming a general ODE in the form of Eq. (2.1) to a special form which can be dealt with directly by the current approach without any iterations).

The special form of ODE that our method requires is

$$y^{(n)}(x) = g(y^{(n-1)}(x), y^{(n-2)}(x), \ldots, y'(x), y(x), x), \tag{2.2}$$

in which the highest derivative term has already been explicitly expressed in terms of other lower order derivative terms. Before we start to describe our solution procedure, we shall first show that any ODE given in the most general form of Eq. (2.1) can always be written in the form of Eq. (2.2) at the expenses of raising the order of the differential equation by one. This can be achieved by simply taking the derivative with respect to x on both sides of Eq. (2.1). Using the chain rule, we have

$$\begin{aligned} \frac{d}{dx}f &= \frac{\partial f}{\partial y^{(n)}}\frac{d}{dx}y^{(n)}(x) + \frac{\partial f}{\partial y^{(n-1)}}\frac{d}{dx}y^{(n-1)}(x) + \ldots \frac{\partial f}{\partial y}\frac{d}{dx}y(x) + \frac{\partial f}{\partial x} \\ &= \frac{\partial f}{\partial y^{(n)}}y^{(n+1)} + \frac{\partial f}{\partial y^{(n-1)}}y^{(n)} + \ldots \frac{\partial f}{\partial y}y' + \frac{\partial f}{\partial x} = 0. \end{aligned} \tag{2.3}$$

Consequently, one can now easily rewrite Eq. (2.3) in the form of Eq. (2.2), assuming $\frac{\partial f}{\partial y^{(n)}} \neq 0$ (this assumption is always correct, or otherwise Eq. (2.1) would have not been an ODE of order n). Of course, the initial

condition a_n is not known. But, we shall assume that a_n can be found from Eq. (2.1) through iterations if necessary.

Therefore, we shall now propose a solution procedure to solve an ODE of the equation (2.2) subject to the initial conditions $y(x_0) = a_0$, $y'(x_0) = a_1$, $\ldots$, $y^{(n-1)}(x_0) = a_{n-1}$.

Like many other numerical approaches, our approach begins also with a discretization of the domain of the independent variable x. Assuming that we want to compute the unknown function values up to $x = X$, where X is a given constant, we first subdivide the domain $[x_0, X]$ into N equal subintervals, each one being of the length of $(X - x_0)/N$. Thus, there are altogether $N+1$ grid points, the collection of which is denoted as the grid set S. Now, if we denote the coordinate of the ith grid point as x_i, where $i = 0, 1, \ldots, N$, our objective is to seek a set of y_i values that approximate the functional values $y(x_i)$.

Our new algorithm is now described as follows.

Suppose at the ith time step, the initial conditions of $y^{(m)}(x_i)$ $(m = 0, 1, \ldots, n-1)$ values at the x_i are either given or calculated in the previous time step. Here, we use superscripts "(m)" to denote the mth order derivative.

To begin our computation, we shall first take a subset of grids, in which a total of $\bar{n} + 1$ grids are selected from the grid set S. Here,

$$\bar{n} = \begin{cases} n+2 & \text{if n is even,} \\ n+1 & \text{if n is odd.} \end{cases} \tag{2.4}$$

$\frac{\bar{n}}{2}$ grids are on the right hand side of x_i and the other $\frac{\bar{n}}{2}$ grids are on the left hand side of x_i. Clearly, the size of this subset (i.e., total number of grids contained in the subset) is pre-determined by the order of the given differential equation. However, its contents (the nodal points) are changing at each time step, and thus we denote the subset corresponding to the ith time step S_i.

For example, for a differential equation of 4th order ($n = 4$), there will be seven grid points placed in the subset S_i at each time step i. If we are calculating the y value at the $5th$ grid point ($i = 5$) in the $5th$ time step, there will be seven grid points placed in the subset S_5. They are $x_j, j = 2, 3, \ldots, 8$. At the beginning, a set of $\frac{\bar{n}}{2}$ fictitious grid points on the left hand side of the x axis (shown in Fig. 1) is created. These grid points are numbered with negative sub-indices, e.g., $x_{-1}, x_{-2} \ldots, x_{-\frac{\bar{n}}{2}}$. Therefore, our grid set S should be modified to contain these fictitious grid points as well. In general, for an nth order equation, $S = \{x_{-\frac{\bar{n}}{2}}, x_{-\frac{\bar{n}}{2}+1}, \ldots, x_0, x_1, \ldots, x_N\}$.

Next, we shall show that using at most $m + 1$ functional values on the discrete grid points contained in the subset, one can approximate the mth-order derivative of y up to the second order of the grid spacing. The formula presented in the following theorem can be regarded as a second-order central difference scheme for the mth-order derivative of y.

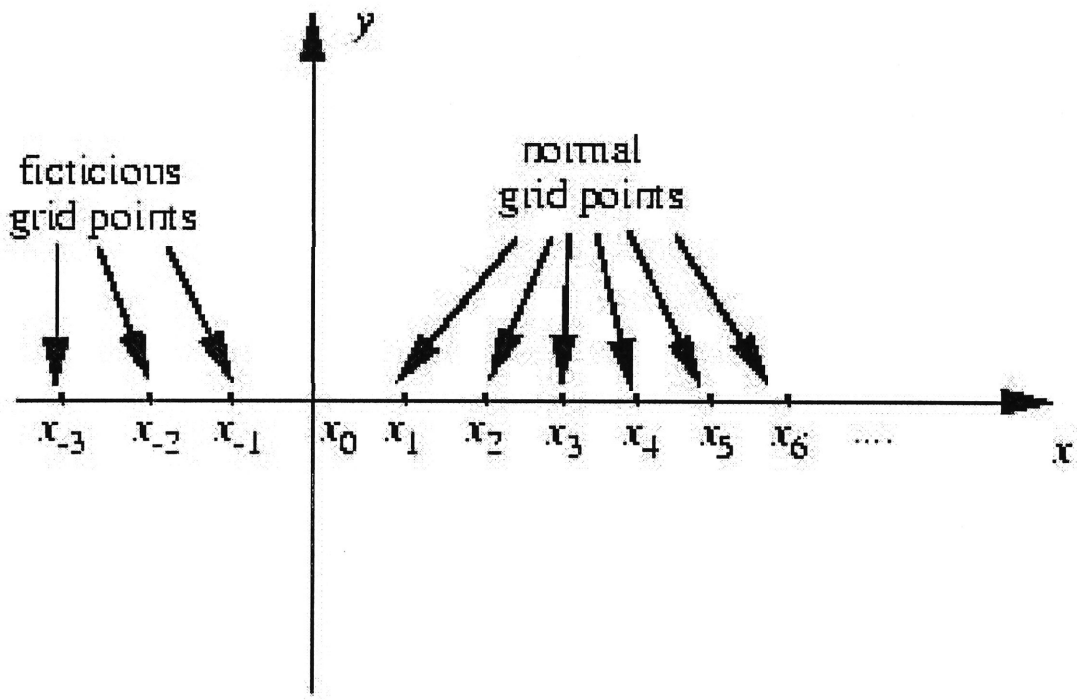

Figure 2.1: Sketch of grid points

Chapter 3

Theorem

If $y(x)$ is a continuous, single-valued function of x with continuous derivatives $y^{(1)}(x), y^{(2)}(x), \ldots$, up to and including $y^{(m+1)}(x)$ in the neighbourhood of x_i, then the mth order derivative of function $y(x)$ at x_i can be approximated by

$$y^{(m)}(x_i) \approx \begin{cases} \frac{1}{h^m} \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} (-1)^{i+\frac{m}{2}} \binom{m}{i+\frac{m}{2}} y_i, & \text{when } m \text{ is even,} \\ \frac{1}{2h^m} \{ y_{\frac{m+1}{2}} + \sum_{i=0}^{m-1} (-1)^i \left[\binom{m}{i} - \binom{m}{i+1} \right] y_{\frac{m-1}{2}-i} - y_{-\frac{m+1}{2}} \}, & \text{when } m \text{ is odd,} \end{cases} \tag{3.1}$$

with the truncation error being of the order of $O(h^2)$, where h is the grid spacing. In Eq. (3.1), y_i is the discrete value of $y(x)$ at x_i and $\binom{n}{k}$ is given by

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}. \tag{3.2}$$

We shall leave the proof of the theorem in the Appendix for the interested readers. However, one should notice that formula (3.1) is very similar to the Newton's formula [10]. In fact, the even-m half is of the

same form as the Newton's formula whereas the odd-m half can be regarded as a geometric average.

As an example to this formula, we have listed 6 equations that arise from approximating the $y^{(6)}(x_0)$.

$$y_0 = h^0 y(x_0),$$

$$y_1 - y_{-1} = 2hy^{(1)}(x_0),$$

$$y_1 - 2y_0 + y_{-1} = h^2 y^{(2)}(x_0),$$

$$y_2 - 2y_1 + 2y_{-1} - y_{-2} = 2h^3 y^{(3)}(x_0), \tag{3.3}$$

$$y_2 - 4y_1 + 6y_0 - 4y_{-1} + y_{-2} = h^4 y^{(4)}(x_0),$$

$$y_3 - 4y_2 + 5y_1 - 5y_{-1} + 4y_{-2} - y_{-3} = 2h^5 y^{(5)}(x_0),$$

$$y_3 - 6y_2 + 15y_1 - 20y_0 + 15y_{-1} - 6y_{-2} + y_{-3} = h^6 y^{(6)}(x_0),$$

The coefficients in the above equation set resemble somewhat those in the Pascal triangle. However, they are not all the same as those in the Pascal triangle (Chinese believe [11] that it should be called Yang Hui triangle as Yang Hui proposed this triangle in 1261, nearly 400 years earlier than Pascal's discovery in 1650). Unlike the forward difference or the backward difference formula, in which all the coefficients are form the Pascal triangle, the coefficients on the left hand side of Eq. (3.3) corresponding to the odd-order derivative are not the same as those in the Pascal triangle whereas those associated with the even-order derivative are the same as those in the Pascal triangle. This is a direct result of the

formation of the central difference formula (3.1). It is worthwhile to point out that Newton Forward-Difference Formula and Newton Backward-Difference Formula can be readily found in many text book (e.g., [8]). However, to the authors' knowledge, a general central difference formula like this one hasn't been proposed before.

Now, using this general central difference formula, we can establish a system of linear equations at each time step. The solution of this set of equations will not only provide us with the numerical solution we are seeking, but also the approximate values of the derivatives needed for the calculation to proceed to the next time step. Here, for the sake of the easiness of describing our algorithm, we shall sometimes take a 4th order ODE as an example in the illustration of our algorithm below.

At the ith time step, let's first take a subset of the grid set S, called it S_i, in which $\bar{n}+1$ grids $\{x_{i-\frac{\bar{n}}{2}}, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_{i+\frac{\bar{n}}{2}}\}$ are taken. Correspondingly, there are $\bar{n}+1$ values of y defined on these grids and we shall denote them as $\{y^i_{i-\frac{\bar{n}}{2}}, \ldots, y^i_{i-1}, y^i_i, y^i_{i+1}, \ldots, y^i_{i+\frac{\bar{n}}{2}}\}$. Notice here that the contents of both subsets are varying with the time step i while the total number of grids included in these two subsets remain unchanged. The reason for adding the superscripts is to emphasize the time dependent nature of these subsets is because of the temporariness of these y values, they are only used within the ith time step and are discarded once the approximation of y_{i+1} and the new "initial conditions" at the $(i+1)$th time step have been determined. In terms of these temporary y values,

we can now rewrite Eq. (3.3) as

$$y_i^i = h^0 y(x_i),$$

$$y_{i+1}^i - y_{i-1}^i = 2hy^{(1)}(x_i),$$

$$y_{i+1}^i - 2y_i^i + y_{i-1}^i = h^2 y^{(2)}(x_i),$$

$$y_{i+2}^i - 2y_{i+1}^i + 2y_{i-1}^i - y_{i-2}^i = 2h^3 y^{(3)}(x_i), \tag{3.4}$$

$$y_{i+2}^i - 4y_{i+1}^i + 6y_i^i - 4y_{i-1}^i + y_{i-2}^i = h^4 y^{(4)}(x_i),$$

$$y_{i+3}^i - 4y_{i+2}^i + 5y_{i+1}^i - 5y_{i-1}^i + 4y_{i-2}^i - y_{i-3}^i = 2h^5 y^{(5)}(x_i),$$

$$y_{i+3}^i - 6y_{i+2}^i + 15y_{i+1}^i - 20y_i^i + 15y_{i-1}^i - 6y_{i-2}^i + y_{i-3}^i = h^6 y^{(6)}(x_i).$$

One should notice that there are altogether seven unknown y values in Eq. (3.4) (i.e., $y_{i-3}^i, y_{i-2}^i, \ldots, y_{i+3}^i$)) defined on the subset S_i at the ith time step. These unknowns can be readily found, provided that we know all the values of $y^{(k)}(x_i), k = 0, 1, 2, \ldots, 6$. In general, for an nth order ODE, the number of equations contained in System (3.4) will be $\bar{n} + 1$ for the $\bar{n} + 1$ unknowns $\{y_{i-\frac{\bar{n}}{2}}^i, \ldots, y_{i-1}^i, y_i^i, y_{i+1}^i, \ldots, y_{i+\frac{\bar{n}}{2}}^i\}$.

Of course, for an nth order initial value problem, we either already know all the initial values of $y^{(k)}(x_i)$ for $k = 1, 2, \ldots, n-1$ from the given initial conditions when we begin our computation with i being set to zero, or all the $y^{(k)}(x_i)$ values for $k = 0, 1, 2, \ldots, n-1$ have been calculated using the information we already obtained from the previous step. With modern symbolic calculation packages (such as Maple) available today, we can then easily calculate the rest of the $y^{(k)}(x_i)$ for $k = n, n+1, \ldots, \bar{n}$, by

repeatedly making use of the differential equation (2.2) itself (i.e., taking the derivative with respect to x in Eq. (2.2) and evaluating the derivative at x_i).

Our new approach completely hinges on this step. Two important points are worthwhile to be mentioned. Firstly, by repeatedly differentiating the original differential equation, rather than trying to "integrate" it, the fact that the equation itself maybe nonlinear poses no further difficulty at all. This is the case is because of this clever usage of the differentiating process to deal with the nonlinearity so that our approach requires no iterations whatsoever! Secondly, this approach would have not been a realistic had symbolic calculation packages not been available since the calculation of the $y^{(k)}(x_i)$ for $k = n, n+1, \ldots, \bar{n}$ values are so tedious that it is such a formidable task no one would really want to tackle by hand.

Once all these derivative values are calculated, the system of equations with the form similar to Eq. (3.4) can be readily solved to obtain the y_k^i values ($k = i - \frac{\bar{n}}{2}, \ldots, i-1, i, \ldots, i + \frac{\bar{n}}{2}$). Then, only one of these y_k^i values is taken as the approximate nodal value and all the rest of these values are only used to determine the initial conditions of the next time step before they are all discarded. Setting $y_{i+1} = y_{i+1}^i$, we shall now proceed to use all the y_k^i values to compute the necessary "initial conditions" at the $(i+1)th$ time step.

Having obtained a set of y_k^i values ($k = i - \frac{\bar{n}}{2}, \ldots, i-1, i, \ldots, i + \frac{\bar{n}}{2}$), one

should realize that again based on the very same difference formulae stated in Eq. (3.4), we can use these values to find the approximate values of the derivatives of $y^{(k)}(x_{i+1})$ up to $k = n-1$. The only difference is that now we shall replace x_i on the right hand side of Eq. (3.4) by x_{i+1}. Correspondingly, all the subscripts on the left hand side are changed too. Let's still use our 4th order equation as an example. After $\{y^i_{i-\frac{\bar{n}}{2}}, \ldots, y^i_{i-1}, y^i_i, y^i_{i+1}, \ldots, y^i_{i+\frac{\bar{n}}{2}}\}$ are computed in the ith step from solving Eq. (3.4), we now replace i in Eq. (3.4) by $i+1$ and obtain

$$
\begin{aligned}
h^0 y(x_{i+1}) &= y^i_{i+1}, \\
2h y^{(1)}(x_{i+1}) &= y^i_{i+2} - y^i_i, \\
h^2 y^{(2)}(x_{i+1}) &= y^i_{i+2} - 2y^i_{i+1} + y^i_i. \\
2h^3 y^{(3)}(x_{i+1}) &= y^i_{i+3} - 2y^i_{i+2} + 2y^i_i - y^i_{i-1},
\end{aligned}
\tag{3.5}
$$

In Eq. (3.5), one should notice that we have deliberately swapped the sides in Eq. (3.4) to indicate that we are now solving for $y^{(1)}(x_{i+1}), y^{(2)}(x_{i+1})$ and $y^{(3)}(x_{i+1})$. Once these new "initial conditions" are determined and stored for the new time step, one can now discard all y^i_k values, update the S_i and move to a new time step.

The above algorithm is repeated in the new time step and march on until the desired time level has been reached.

To illustrate our new approach, we shall give three numerical examples in the next section. We should first briefly point out that there are two ways of handling the last few steps of the computation. One way is of

course to create $\frac{\bar{n}}{2}$ fictitious grids beyond x_N to facilitate the calculation as at the beginning of our algorithm. Alternatively, one could also directly use those computed y_k^i values in the last few steps, rather than discard all but one of them as we did in all the previous time steps. The latter approach does not require the creation of the extra fictitious grids beyond x_N, since at the time step $i = N - \frac{\bar{n}}{2}$, we can simply assign

$$y_{N-\frac{\bar{n}}{2}+1} = y^i_{N-\frac{\bar{n}}{2}+1},$$

$$y_{N-\frac{\bar{n}}{2}+2} = y^i_{N-\frac{\bar{n}}{2}+2},$$

$$\vdots$$

$$y_N = y^i_N.$$

In all the examples presented in the next section, we used the latter approach.

Chapter 4

Results

To demonstrate the accuracy of the proposed approach, we shall present three numerical examples in this section. In some test cases (e.g., Examples 1 and 2), the nonlinear differential equations are specially constructed, so that analytical solutions can be found (in fact, it was constructed in an inverse way. i.e., we constructed the nonlinear equation based on a solution). Using these analytical solutions, we can check the relative and absolute errors for each case.

Example 1:

In the first example, we solve the nonlinear ODE

$$y^{(4)}(x) = 0.09\left(\frac{\left(y^{(2)}(x)\right)^2 y'(x)y(x)}{y^{(3)}(x)} + y^{(2)}(x)y(x)^2\right), \tag{4.1}$$

subject to the following initial conditions

$$y(0) = 1,$$

$$y'(0) = -0.1,$$

$$y^{(2)}(0) = 0.02,$$

$$y^{(3)}(0) = -0.006.$$

The exact solution is $y = \frac{10}{10+x}$, which can be readily verified.

Table 4.1: Maximum absolute and relative errors in Example 1

Grid Spacing h	0.1	0.2	0.4
Max. Abs. Error	0.339×10^{-3}	0.119×10^{-2}	0.265
Max. Rel. Error (%)	0.068	0.237	0.265

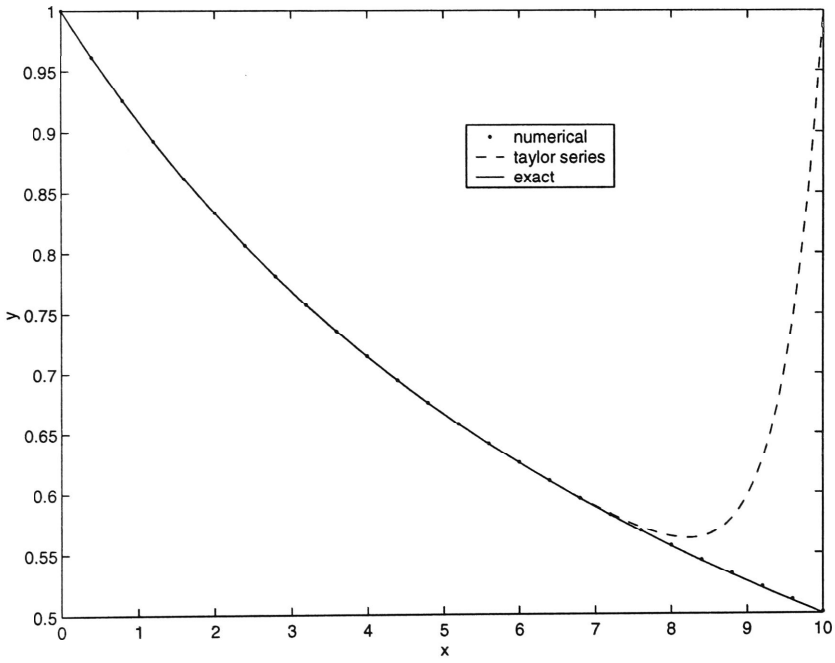

Figure 4.1: Comparison of the numerical and the exact solution of Eq. 4.1

If we take the steplength to be 0.4 and compute the results between $x \in [0, 10]$, the difference between the numerical solution and the exact solution is hardly distinguishable on the graph as shown in Fig. 4.1. With this rather crude grid spacing, the numerical results are already accurate to the second decimal place and the maximum relative error is about

0.265% as shown in Table 1. If we further reduce the grid spacing to 0.2, the maximum relative error only changed slightly to 0.237%. However, if the grid spacing is further halved to 0.1, the maximum relative error is substantially reduced to 0.068%, about a quarter of the maximum relative error with $h = 0.4$. With $h = 0.1$, numerical results accurate to the third decimal place were obtained; a further reduction of grid spacing resulted in a higher accuracy until the machine accuracy was reached.

Example 2:

Our second example is very similar to the first one except that we set $x_0 = 1.5$ rather than zero as in the first example. Also, non-constant coefficients of sinusoidal functions are introduced to test our algorithm in handling ODEs with variable coefficients.

The differential equation now is a 5th order equation

$$\begin{aligned} y^{(5)}(x) = &-4\sin(x)\,(1+x)\left(\sin(x) - y^{(4)}(x)\right) - 12y^{(3)}(x)y^{(2)}(x) \\ &+\tfrac{12\cos(x)}{(1+x)}\left(y'(x) - \cos(x)\right) + y(x) + 6\sin(2x) \\ &+\cos(x) - \sin(x) - \ln(1+x), \end{aligned} \tag{4.2}$$

and the initial conditions are

$$y(1.5) = 1.9137857,$$

$$y'(1.5) = 0.470737,$$

$$y^{(2)}(1.5) = -1.157494987,$$

$$y^{(3)}(1.5) = 0.05726279,$$

$$y^{(4)}(1.5) = 0.84389498.$$

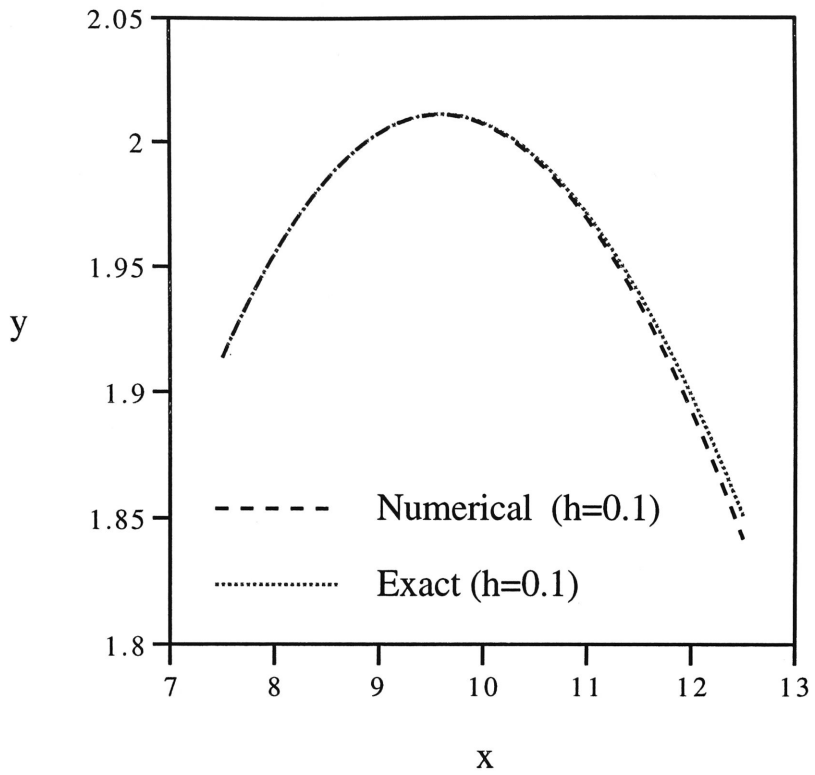

Figure 4.2: The comparison of the numerical and the exact solution of Eq. 4.2

One can easily verify that $y(x) = \sin(x) + \ln(1 + x)$ is the analytical solution satisfying the differential equation and all the initial conditions.

Our numerical results with a grid spacing of $h = 0.02$ are shown in Fig. 4.2 together with the analytical solution. Clearly, numerical errors become slightly larger when x increases. With this grid spacing, the maximum relative error occurs near $x = 2.5$, where one can see from Fig. 4.2 that there are indeed some differences between the two curves. However, the maximum error is only about 0.5%, which is generally acceptable in most of engineering calculations.

Example 3:

Having demonstrated how our new approach is applied to solve nonlinear ODEs in which the highest derivative term can be explicitly expressed in terms of other lower order derivative terms as shown in the previous two examples, we now demonstrate how our approach can be used in case when the highest derivative term can not be expressed in terms of other lower order derivative terms. As discussed in Sec. 2 already. The only extra care one needs to take is to work out the initial value of the highest order derivative through iterations. If this can be done in a fairly straightforward way, then the rest remains the same as the standard procedures described in Sec. 2.

Let's try to solve

$$sin^{-1}y^{(4)}(x) - (y^{(4)}(x))^2 + y'(x)(y(x))^2 - \cos(x)(y^{(2)}(x))^3 y(x) = 0, \qquad (4.3)$$

subject to the following initial conditions

$$y(1) = 0.841471,$$

$$y'(1) = 0.540302,$$

$$y''(1) = -0.841471,$$

$$y^{(3)}(1) = -0.540302.$$

Clearly, the highest derivative of the unknown function $y^{(4)}(x)$ in Eq. (4.3) cannot be written explicitly in terms of all the other lower-order derivative terms. As stated in Sec 2, to use the current method, we first

change the differential equation into a form, in which the highest-order derivative term can be explicitly expressed in terms of other lower order derivative terms. This is achieved by differentiating both sides of Eq. (4.3) with respect to x. After some algebraic manipulations, the new equation can be written as

$$y^{(5)} = \frac{\sqrt{1-(y^{(4)})^2}}{1-2y^{(4)}\sqrt{1-(y^{(4)})^2}}\{\cos(x)(y'')^2[y'y''+3yy^{(3)}]-\sin(x)y(y'')^3-y^2y''-2y(y')^2\} \tag{4.4}$$

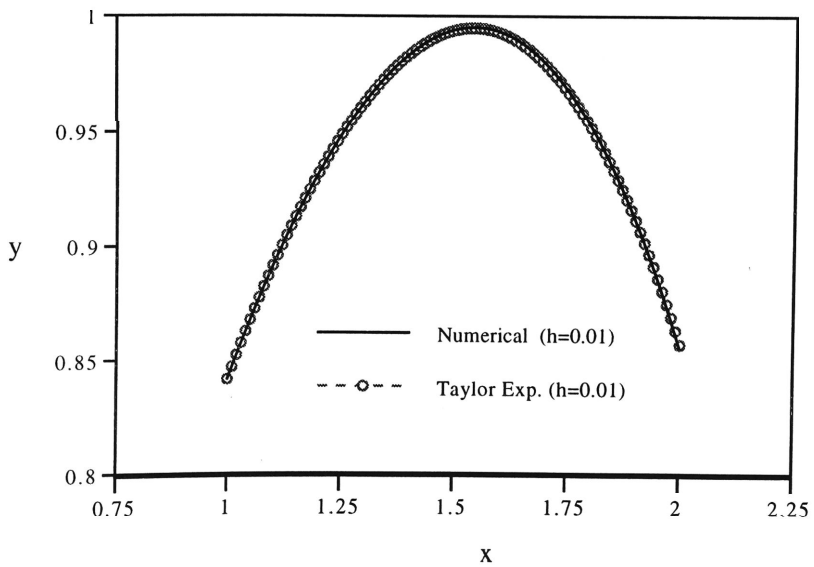

Figure 4.3: Comparison of the numerical solution and the solution based on the Taylor's series expansion method

We have raised the order of the equation to be solved by one in order to change it into the form that the currently proposed method should begin with. The only extra step we need to carry out before our approach can be used is to figure out the $y^{(4)}(1)$ value, which is found through inserting

all initial values into Eq. (4.3) and solving for the unknown $y^{(4)}(1)$ that satisfies

$$sin^{-1}y^{(4)}(1) - (y^{(4)}(1))^2 = y'(1)(y(1))^2 + \cos(1)(y^{(2)}(1))^3 y(1) = -0.65346396. \quad (4.5)$$

This can be easily done with an iteration scheme such the Newton's method (there is a built-in solver based on the Newton's method in Maple. which is the one we used). Once this initial value for $y^{(4)}$ is worked out (for the current example, we used the built-in solver based on the Newton's method in Maple and found that $y^{(4)}(1) = -0.4421365$), the rest of the solution procedure is identical to those in Ex. 1 and Ex. 2.

Since there is no analytical solution for this problem, we have compared the solution obtained with our new approach with that based on a Taylor's series expansion method built in Maple [12]. Shown in Fig. 4.3 is a comparison of our results with those obtained by directly using the Taylor's series expansion method built in Maple. With $h = 0.01$, the difference between the two is hardly noticeable. In fact, when $h = 0.04$, the difference between the two was hardly noticeable already but the maximum relative difference between the two was about 1.0%. When we further reduced the grid spacing h from 0.04 to 0.01, the maximum relative difference between the two became 0.2%.

It is worth pointing out that the built-in approach in Maple based on the Taylor's series expansion method sometimes doesn't converge during

the iterations for a nonlinear problem whereas our approach seems to be more versatile; we haven't encountered a single case that ours failed (simply because we don't need to iterate at all). This is also the particular reason that we chose this equation as our last example because for this case, the built-in approach in Maple based on the Taylor's series expansion works too so that we could have a comparison. In short, in terms of reliability, our approach seems to be better than the Taylor's series expansion method built in Maple.

Chapter 5

Conclusion

In this thesis, we have demonstrated an innovative approach for solving initial value problems governed by non-linear ordinary differential equations. The proposed new approach is based on the repeated use of the differential equation itself and thus must be used in conjunction with a modern symbolic calculation package, such as Maple or Mathematica. The method is particularly useful when the nonlinear ordinary differential equations to be solved are of order higher than one. Through our examples, we have shown that, if the highest derivative term can be written explicitly in terms of other terms, there is no need of nonlinear iterations at each time step, while iterations are only needed before the time marching begins when the highest derivative term can not be written explicitly in terms of other terms.

Chapter 6

Appendix

Here we should prove the Theorem in Section 3. For the sake of simplicity, we shall set $i = 0$ and discuss two cases, i.e., when m is even or m is odd, separately.

When m is even, applying the Taylor theorem, one can express grid nodal values y_k $(k = -\frac{m}{2}, \ldots, \frac{m}{2})$ in terms of Taylor series as

$$\begin{aligned} y_{-\frac{m}{2}} &= y(x_0 - mh/2) = y_0 - \frac{m}{2}hy_0' + \frac{(\frac{m}{2}h)^2 y_0''}{2!} - \ldots + \frac{(\frac{m}{2}h)^m y_0^{(m)}}{m!} - \frac{(\frac{m}{2}h)^{m+1} y_0^{(m+1)}}{(m+1)!} \\ &\quad + O(h^{m+2}), \end{aligned}$$

$$\vdots$$

$$y_{-2} = y(x_0 - 2h) = y_0 - 2hy_0' + \frac{(2h)^2 y_0''}{2!} - \ldots + \frac{(2h)^m y_0^{(m)}}{m!} - \frac{(2h)^{m+1} y_0^{(m+1)}}{(m+1)!} + O(h^{m+2}),$$

$$y_{-1} = y(x_0 + h) = y_0 - hy_0' + \frac{h^2 y_0''}{2!} - \ldots + \frac{h^m y_0^{(m)}}{m!} - \frac{h^{m+1} y_0^{(m+1)}}{(m+1)!} + O(h^{m+2}),$$

$$y_0 = y(x_0) = y_0,$$

$$
\begin{aligned}
y_1 &= y(x_0+h) = y_0 + hy_0' + \frac{h^2 y_0''}{2!} + \ldots + \frac{h^m y_0^{(m)}}{m!} + \frac{h^{m+1} y_0^{(m+1)}}{(m+1)!} + O(h^{m+2}), \\
y_2 &= y(x_0+2h) = y_0 + 2hy_0' + \frac{(2h)^2 y_0''}{2!} + \ldots + \frac{(2h)^m y_0^{(m)}}{m!} + \frac{(2h)^{m+1} y_0^{(m+1)}}{(m+1)!} + O(h^{m+2}), \\
&\vdots \\
y_{\frac{m}{2}} &= y(x_0+mh/2) = y_0 + \frac{m}{2} hy_0' + \frac{(\frac{m}{2}h)^2 y_0''}{2!} + \ldots + \frac{(\frac{m}{2}h)^m y_0^{(m)}}{m!} + \frac{(\frac{m}{2}h)^{m+1} y_0^{(m+1)}}{(m+1)!} \\
&\quad + O(h^{m+2}).
\end{aligned}
$$

Now, if we multiply both sides of the ith equation by $(-1)^{i+\frac{m}{2}} \binom{m}{i+\frac{m}{2}}$ and then add all the equations up, the sum on the left hand side of the resulting equation is

$$\sum_{i=-\frac{m}{2}}^{m/2} (-1)^{i+\frac{m}{2}} \binom{m}{i+\frac{m}{2}} y_i,$$

while after the cancellations of the coefficients for all the terms except that of the $O(h^m)$ term and those of $O(h^{m+2})$ terms, the right hand side becomes

$$h^m y_0^{(m)} + O(h^{m+2}).$$

In other words, the resulting equation becomes

$$\sum_{i=-\frac{m}{2}}^{m/2} (-1)^{i+\frac{m}{2}} \binom{m}{i+\frac{m}{2}} y_i = h^m y_0^{(m)} + O(h^{m+2}),$$

or

$$y_0^{(m)} = \frac{1}{h^m} \sum_{i=-\frac{m}{2}}^{m/2} (-1)^{i+\frac{m}{2}} \binom{m}{i+\frac{m}{2}} y_i + O(h^2),$$

which is the first half stated in Eq. (3.1).

When m is odd, the proof is very similar. Instead of having $m+1$ equations, we now have $m+2$ equations for y_k with $k = -\frac{m+1}{2}, \ldots, \frac{m+1}{2}$.

Then, in stead of multiplying the ith equation by $(-1)^{i+\frac{m}{2}}\binom{m}{i+\frac{m}{2}}$, we shall multiply the 1st equation (the one that corresponds to $y_{-\frac{m+1}{2}}$) by -1 and the last equation (the one that corresponds to $y_{\frac{m+1}{2}}$) by 1. Starting from the 2nd equation, the $(i+2)$th equation ($i = 0, 1, ...m-1$) is multiplied by $(-1)^i\left[\binom{m}{i} - \binom{m}{i+1}\right]$. The rest of the proof is identical to that shown above and is thus omitted here.

Bibliography

[1] Menahem Friedman and Abraham Kandel, Fundamentals of Computer Numerical Analysis, (1994), pp.409-422

[2] Shoichiro Nakamura, Applied Numerical Methods With Software, Prentice Hall, New Jersey (1991), pp. 303-308

[3] Fox, L. and Mayers, D. F., Numerical Solution of Ordinary Differential Equations, *Chapman and Hall*, 1987.

[4] Quinney, D., An Introduction to Numerical Solution of Differential Equations, *Research Studies Press*, 1987.

[5] Horn, M.K., Fourth- and fifth-order, scaled Runge-Kutta algorithms for treating dense output, *SIAM J. Numer. Anal.*, 1983, 20,558-568.

[6] Nosett, S. P. and Wanner, G., Perturbed Collocation and Runge-Kutta methods, *Numer. Math.*, 1981, 38,193-208.

[7] Brynjulf, O. and Zennaro, M., Continuous Explicit Runge-Kutta methods, *Computational Ordinary Differential Equations*, edited by J. R. Cash ad I. Gladwell, 1992, 97-105.

[8] Richard L. Burden and J. Douglas Faires, Numerical Analysis, PWS Boston (1993), pp.248-252, 254-258, 265-268

[9] A.S. Cakmak, J.F. Botha and W.G. Gray, Computational and Applied Mathematics for Engineering Analysis, Springer-Verlag, Berlin (1987), pp. 183-234

[10] Gene H. Golub and James M. Ortega, Scientific Computing and Differential equations (1992), pp.42-61, 179-191

[11] *Mathematics Handbook*, pp27, 1979 (in Chinese).

[12] W. Ellis, E. Johnson, W. Lodi and D. Schwalbe, Maple V, *Brooks/Cole Publishing Company*, (1997).

Betta Book Binding Printing
M & D Morrisey 4261 2998
26 Fields Street
Kanahooka NSW 2530